设计实验室 6　　6.01 –Fall 2011

庞大的关注者群体

目标： 我们已经看到，简单的*比例控制器*在引导机器人沿着墙壁跟随方面表现不佳。在本实验中，我们研究了针对 **WallFollower** 的两个改进控制器：

- 一个延迟加比例控制器，它取决于与墙壁的前一距离以及当前距离
- 一个角度加速度控制器，它取决于当前与墙壁的角度以及当前的距离

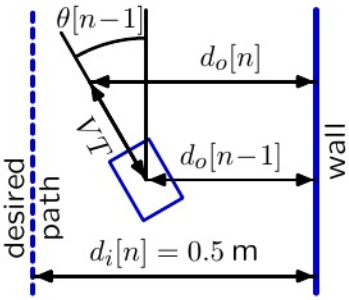
1 引言

资源：本实验应与搭档合作完成。每组搭档应配备一个**机器人**以及一台可靠的运行 SOAR 的实验室**笔记本**电脑或个人笔记本电脑。请使用 athrun 6.01 getFiles 命令获取本实验所需的文件，这些文件位于 Desktop/6.01/designLab06 中，或者从课程网页获取软件分发包。分发包中的相关文件包括：

- delayPlusPropBrainSkeleton.py：延迟加比例控制器的脑模板。
- anglePlusPropBrainSkeleton.py：角度加比例控制器的大脑模板。
- designLab06Work.py：包含用于创建系统功能和优化的导入语句的文件。
- 另请参阅课程笔记的第 **5 章**。

一定要把你的所有代码和图表寄给你的搭档。你们每个人都需要把副本带到面试现场。

上周，我们使用**比例**控制器来使机器人平行于墙壁移动，同时尽量保持与墙壁的期望距离恒定。前向速度V设置为0.1 m/s，角速度ω[n]与误差信号e[n]成正比，误差信号e[n]是期望距离d_i[n]和当前距离d_o[n]之间的差值。不幸的是，没有比例常数k的值能给出良好的性能。大的k值会导致快速振荡，而小的k值会导致较大的误差，尤其是当机器人的初始角度不与墙壁平行时。在本实验中，我们将开发



两种新型控制器以实现更出色的性能；第一种
第一个控制器取决于之前与墙壁的距离以及当前的距离，而第二个控制器取决于当前与墙壁的角度以及当前的距离。

Some of the software and design labs contain the command athrun 6.01 getFiles. Please disregard this instruction; the same files are available on the 6.01 OCW Scholar site as a .zip file, labeled Code for [Design or Software Lab number].

2 《追忆似水年华》

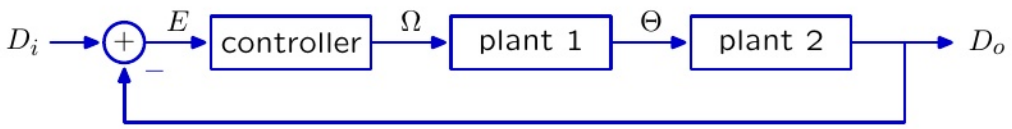
目标： 按照以下步骤制作一个控制器，该控制器要同时考虑与墙壁的前一次距离以及当前距离：

- 构建一个延迟加比例控制器 - 植物 - 传感器的模型系统，使用“**系统功能**”类。
- 利用该模型来获取最佳收益。
- 为机器人构建一个相应的控制器。
- 针对各种增益在模拟中进行测试。
- 在一个真正的机器人上进行测试。

通过以比我们在上一次实验中更复杂的方式处理误差信号 E，可以制造出更好的跟踪墙壁控制器。例如，我们可以使用误差当前值和前值的某种组合来调整角速度。

$$\omega[n] = k_1 e[n] + k_2 e[n-1].$$

我们将这个控制器称为“延迟加比例”。该系统仍然与上周的那个系统形式相同：



代表机器人移动的子系统（植物 1 和植物 2）以及传感器（建模为电线）是相同的。只有控制器发生了变化。

详细指导：

*自我*检查 *1.* 画一个控制器的方框图。

记住，ω 是小写的 Ω 。

2.1 模型

步骤 1. 使用 Python 的 **SystemFunction 类**对机器人具有延迟加比例控制器时整个系统的行为进行建模和分析，如下所示。

- 在 idle 中打开文件“designLab06Work.py”，并将其用于本实验的这一部分。
- 编写一个 Python 程序 delayPlusPropModel，该程序以增益 k1 和 k2 作为输入，并返回一个 SystemFunction，用于描述当机器人具有延迟加比例控制器时系统的行为。您可以假设 T = 0.1 秒，V = 0.1 米/秒。

  您可以使用 sf.FeedforwardAdd 和 sf.FeedforwardSubtract 这两个组合子来进行系统的简单相加，也可以使用本课程网页**参考材料**中**软件文档**部分的 **sf 模块**页面中的任何其他 sf 组合子。它们都有相关**文档说明**。

  利用上周问题 **Wk.5.3.4** 的答案来帮助解决这个问题。

> **第 6 周 1.1.1. 第一部分** 将**您**的 delayPlusPropModel 代码**输入**到辅导系统中。

2.2 选择收益

我们想要选择k1和k2的值，以获得最佳稳定的性能。正如我们在课上看到的，通过**减小主导极点**的**幅**值可以提高收敛速度。让我们先考虑单个增益的情况，**即**你上周建模的系统。你可以构建一个函数f(k)来计算系统主导极点的幅值。现在，你需要找到使这个函数最小值的k值。

寻找一个函数的最小值是一个优化过程，可以使用 **lib601 优化模块**中的例程来完成，如**作业 2** 中所述。为了您的方便，以下重现了该文档。

给定一个函数f(x)，我们如何找到使得f(x)最小的x值？如果f是可微的，我们可以通过求导、令导数为0并求解x来实现。当函数f很复杂，可能存在多个最小值，或者希望扩展到多变量函数时，这种方法就会变得棘手。对于不可微的函数（如涉及最大值或绝对值的函数），根本没有简单的数学方法。在一维中，如果我们知道一个可能包含最小值的x的范围，我们可以在该范围内合理地采样不同的x值，计算每个x值上的f(x)，并返回使f(x)最小的采样x值。

程序 optimize.**optOverLine** 实现了这一功能。它是 **lib601 optimize 模块**的一部分，调用方式如下：

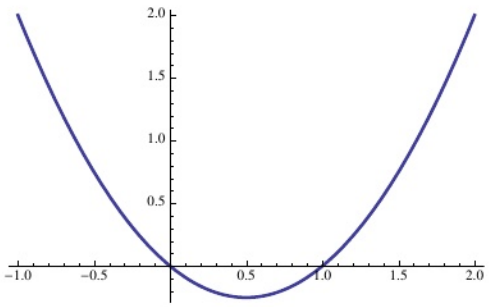
optimize.optOverLine（目标函数、最小 x 值、最大 x 值、x 步长数、比较）

- 目标：一个接受单个数值参数（在我们的示例中是 x）并返回一个数值的程序（在我们的示例中是 f(x)）
- xmin，xmax：自变量的一组取值范围

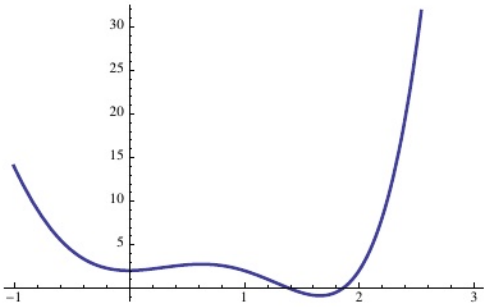
· numXsteps：在该范围内测试多少个点
· compare：一个可选的比较函数，默认值为 operator.lt，即 Python 中的小于运算符 < 。这意味着如果您未指定 compare 参数，该过程将返回使目标最小化**的值**。

该程序返回一个元组（bestObjValue，bestX），其中包含目标的最佳值（在我们的示例中为(f(x^*)）以及与之对应的 x 值（在我们的示例中为(x^*in）。

Check Yourself 2. Here are graphs of two functions. Use `optimize.optOverLine` to find the minimum of one of them. The function f has a minimum at $x = 0.5$ of value -0.25; the function h has a minimum at $x = 1.66$ with value -0.88.



$f(x) = x^2 - x$



$h(x) = x^5 - 7x^3 + 6x^2 + 2$

步骤 2. 考虑 k1 的四个不同值：10、30、100 和 300。对于每个 k1 的值，使用 optimize.optOverLine 来确定使最不稳定极点的大小最小化的 k2 的值。

一定要测试k2的正负值，确保测试的范围足够大，并且最终采样粒度至少为0.1。与其设置一次长距离的最小化运行，范围宽且粒度小，不如先以粗粒度开始，找到正确的粗略值，然后在其周围进行更精细的搜索。提示：k2的绝对值不应大于k1的绝对值。

我们鼓励您为此目的定义并使用包含在设计Lab06Work.py 中的 bestk2 过程存根。

k1	k2	magnitude of dominant pole
10		
30		
100		
300		

第 6 周 1.1.1. 第 **2 部分** 输入您为上述每个 k1 值找到的 k2 值以及相关主导极点的幅值。

2.3 大脑

步骤 3. 通过编辑 delayPlusPropBrainSkeleton.py 中的 WallFollower 状态机类**来**实现延迟加比例控制器。**仔细思考在第一个时间步长中您想要输出什么**。

和之前一样，大脑有两个部分串联连接。第一部分是Sensor类的实例，它实现了一个状态机，其输入是一系列**io.SensorInput**类的实例，输出是到墙壁的垂直距离。垂直距离是通过**sonarDist模块**中的getDistanceRight函数使用三角测量计算得出的（假设墙壁是局部直的）。如果传感器6和7都击中了墙壁，那么这个值就相当准确。如果只有其中一个击中了，那么它就不那么准确。如果两个传感器都没有击中墙壁，那么它返回1.5。Sensor类的代码已经提供。

大脑的第二部分是 WallFollower 类的一个实例。您应该提供代码，使 WallFollower 类实现一个状态机，其输入是到墙壁的垂直距离，输出是 **io.Action 类**的一个实例。仔细考虑您要在机器的状态中存储什么，以及如何初始化 startState 。

大脑的设置是这样的：每当您点击“**停止**”时，就会显示一个图表，展示到右侧墙壁的垂直距离如何随时间变化。要保存此图表，请按照我们网页上的说明截取屏幕截图。**一旦您重新加载大脑，此图表就会消失**。

步骤4. 运行你的大脑在世界墙测试程序TestWorld.py中。确定系统的行为如何受到控制器增益k1和k2的影响。只使用你在前一步确定的增益。注意大脑打印出的距离值。保存绘图以说明你优化后的增益对对每对k1和k2的性能的影响。为这些文件选择名称，以便你能记住生成每个文件所使用的参数。

保留好你的口头面试的文件。

不要为了试图让控制器更好地工作而改变它！相反，要尝试去理解可能出现的不同种类的故障，以及它们如何取决于增益或初始条件的选择。

Check Yourself 3. Which of the four gain pairs work best in simulation?

`k1 =` `k2 =`

Which gains cause bad behavior?

第 5 步。将您的实验笔记本电脑连接到机器人上。您可能需要一根长以太网电缆（或者请工作人员帮忙切换到无线网络）。

走到房间边缘或者到走廊上去，找一面长长的（至少两张气泡膜包装的木板）墙来操作。

在一个真正的机器人上运行你的大脑。拿一个测量棒，并确保让机器人在相同的初始条件下启动（距离墙壁 0.5 米，向左旋转π/8 弧度），就像在模拟器中一样。留意大脑打印出的距离值。

为您的每一对计算得出的收益保存一个图表。保留好你的口头面试的文件。

自我检测 4. 在机器人中，这四种增益对中哪一种效果最佳？

k1 等于 k2 等于

最佳收益与模拟中的收益相同吗？
哪些收益会导致不良行为？

核对 1 。

第 6 周 1.2：向一名工作人员展示模拟和真实机器人运行的绘图，并讨论它们之间的关系。对于每对增益，机器人的行为与主导极点的幅值有何关系？

解释一下您如何选择控制器的初始状态。

确保双方都有这些文件。

3 如果我们正朝着正确的方向前进，我们所要做的就是继续前行。

目标：

使用以下步骤制作一个控制器，该控制器取决于当前与墙壁的角度以及当前的距离：

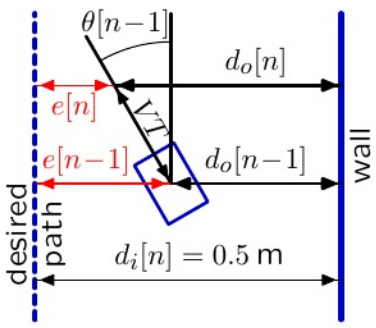
- 构建一个角度加比例控制器 - 植物 - 传感器的模型系统，使用“系统功能”类。
- 利用该模型来获取最佳收益。
- 为机器人构建一个相应的控制器。
- 针对各种增益在模拟中进行测试。
- 在一个真正的机器人上进行测试。

为什么上一节中延迟加比例控制器比上周的比例控制器表现更好？唯一的区别在于对 e[n - 1] 的访问。那么，为什么旧信息会有所帮助呢？一个相关的问题是，为什么 $-k_2$ 的最佳值只比 k_1 的最佳值略小一点。

思考延迟加比例控制器的一种更好方式是将其视为比例加微分控制器，这里的微分是指一阶差分。正如我们将在下文看到的，如果我们

以差值而非延迟的形式重新参数化增益，那么增益之间的关系（$k_2 = -k_1 + \epsilon$）就不再神秘了。

对于这个特定的问题，差值 e[n] - e[n - 1] 可以从角度 θ[n - 1] 的角度以图形方式进行解释（见右图）。因此，延迟加比例控制器可以根据机器人的位置和角度来确定下一个角速度（其输出）。然而，请注意，关于位置的信息是针对时间 n 的，而关于角度的信息是针对时间 n - 1 的。



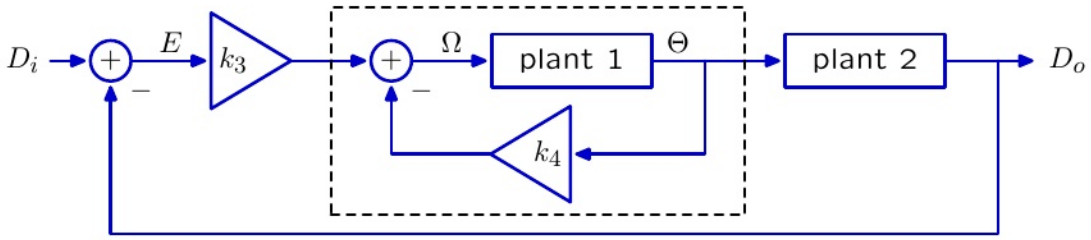
我们的机器人不仅仅有一个传感器——实际上它有八个声纳，以略有不同的角度排列——这样我们就可以直接测量角度了。如果一个控制系统有这些传感器，它的工作效果会如何?
关于位置和角度的最新信息？我们可以通过分析角度加比例控制器来回答这个问题:

$$\omega[n] = k_3(d_i[n] - d_o[n]) + k_4(\theta_d - \theta[n])$$

其中$\theta_d = 0$ 表示期望的角度（相对于墙壁测量），因此

$$\omega[n] = k_3 e[n] - k_4 \theta[n]$$

如下图所示的方框图。



请注意，该控制器有一个反馈回路来控制角度Θ，而Θ位于第二个反馈回路内，用于控制距离D_0。

详细指导:

3.1 模型

第 6 步。在 designLab06Work.py 中，编写一个 Python 程序 anglePlusPropModel，该程序以增益 k3 和 k4 作为输入，并返回一个描述角度加比例控制的系统的 SystemFunction 。假设 T = 0.1 秒，V = 0.1 米/秒。

> **第 6 周 1.3 第一部分** 将您的 anglePlusPropModel 代码输入到辅导系统中。

第 7 步。对于 k3 等于 1、3、10 和 30 的情况，确定使角度加比例系统中最不稳定极点的幅值最小化的 k4 的值。

k3	k4	主导极的幅度
1		
3		
10		
30		

第 6 周 1.3 部分 2 输入您为上述每个 k3 值找到的 k4 值以及相关主导极点的幅值。

3.2 大脑

第 8 步。通过编辑 anglePlusPropBrainSkeleton.py 中的 WallFollower 状态机类来实现比例加角度控制器。

和以前一样，大脑有两个部分串联连接。第一部分是 Sensor 类的一个实例，它实现了一个状态机，其输入是一系列 SensorInputs 的实例，输出是一系列右**墙垂直距离和与右墙夹角**的**对**。

大脑的第二部分是 WallFollower 类的一个实例。您应该提供代码，使 WallFollower 类实现一个状态机，其输入是**到右侧**墙壁的**垂直距离和与墙壁**的**角度**这一**对**值，其输出是该类 io.Action 的一个实例。

请注意，如果声纳 6 或 7 超出范围，那么我们无法计算角度，并且传感器机器的输出中的第二个分量将为空。当这种情况发生时，您**的**大脑应该**将**角速度设置为 0 。

第 9 步。使用 wallTestWorld.py 世界在 soar 模拟器中测试您的大脑是否正常工作。为您的每对计算增益保存一个绘图。保留好你的口头面试的文件。

*自我*检查 5. 在模拟中，这四种增益对中哪一种效果最佳?

k3 等于 k4 等于

哪些收益会导致不良行为?

第 10 步走到房间边缘或者到走廊上去，找一面长长的（至少两张气泡膜包装的木板）墙来操作。

在一个真正的机器人上运行你的大脑。拿一个测量棒，并确保让机器人在相同的初始条件下启动（距离墙壁 0.5 米，向左旋转π/8 弧度），就像在模拟器中一样。留意大脑打印出的距离值。为每一对计算得出的增益值保存一个绘图。保留好你的口头面试的文件。

自我检测 6. 在机器人中，以下四种增益对哪一种效果最佳?

k3 等于 k4 等于

最佳收益与模拟中的收益相同吗?
哪些收益会导致不良行为?

核对 2。 **第 6 周 1.4**：向一名工作人员展示模拟和真实机器人运行的绘图。
并讨论它们之间的关系。机器人的行为与.....有何关联?
对于每对增益，主导极点的幅度是多少?

哪种控制器（延迟加比例或角度加比例）
形式更好? 解释原因。

确保双方都有这些文件。

6.01SC 《电气工程与计算机科学导论》 2011 年春季